CONTENTS

		Page	
1.	Introduction	9	1/A9
2.	The Facility Model	10	1/A10
3.	The Path Problem	10	1/A10
4.	The Solution	11	1/A11
5.	An Example	12	1/A12
6.	Calling POST	14	1/A14
7.	Solving the Example with POST	17	1/B3
8.	Description of POST	18	1/B4
9.	Subroutine Performance	21	1/B7
	References	23	1/B9
	POST Listings (microfiche)	back cover 26	1/B12

FIGURES

Figure			
1	Transition Point P_3	11	1/A11
2	A Weighted Facility Digraph	13	1/A13

TABLES

Table			
1	POST Performance	21	1/B7
2	Range of POST Run Times with Different Given Target Orderings	22	1/B8

JUN 3 1980

Item 105-H-11

Y3.N88:25/1226

COMPLETED ORIGINAL

NUREG/CR-1226
SAND80-0058
RS

POST: A Subroutine for Path Ordering of Sabotage Targets

Bernie L. Hulme, Christine A. Morgan

Sandia National Laboratories

F 2900-Q(3-80)

Prepared for
U. S. NUCLEAR REGULATORY COMMISSION

26

NOTICE

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, or any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for any third party's use, or the results of such use, of any information, apparatus, product or process disclosed in this report, or represents that its use by such third party would not infringe privately owned rights.

The views expressed in this report are not necessarily those of the U. S. Nuclear Regulatory Commission

Available from

GPO Sales Program
Division of Technical Information and Document Control
U.S. Nuclear Regulatory Commission
Washington, D.C. 20555

and

National Technical Information Service
Springfield, Virginia 22161

NUREG/CR-1226
SAND80-0058
RS

POST: A Subroutine for Path Ordering of Sabotage Targets

Bernie L. Hulme
Christine A. Morgan

Date Published: February 1980

Sandia Laboratories
Albuquerque, New Mexico 87185
operated by
Sandia Corporation
for the
U.S. Department of Energy

Prepared for
Division of Safeguards, Fuel Cycle and Environmental Research
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, D.C. 20555
Under Memorandum of Understanding DOE 40-550-75
NRC FIN No. A1060

BLANK PAGE

ABSTRACT

POST is a subroutine which a safeguards analyst may use to find multiple-target sabotage paths through a fixed-site facility. This subroutine accepts the same weighted digraph facility model as does ADPATH, the single-target adversary path code. Given a list of potential starting nodes and a list of one set of target nodes together with their respective guard-response times, POST computes an adversary's minimum interruption probability path. This path begins at one of the starting nodes and passes through all of the given target nodes in such a way as to minimize the adversary's probability of being detected while the guards still have time to respond to an alarm and interrupt the sabotage.

BLANK PAGE

BLANK PAGE

POST: A Subroutine for Path Ordering of Sabotage Targets

1. Introduction

Up to now safeguards pathfinding codes [2,3,4] have concentrated on single-target paths. This is appropriate because theft of nuclear materials typically involves visting only one target and a worst-case sabotage scenario involves the simultaneous attack of multiple teams each having only one target. There remains, however, considerable interest in studying the case of a single sabotage team having more than one target to visit. This problem arises not only when just one team is postulated but more generally when one assumes that there are more targets than teams. In the latter case, the safeguards analyst will want to consider various partitions of the target set, there always being at least one team with more than one target to visit.

The objective function in POST is the same as in ADPATH and MINDPT, i.e., to minimize interruption probability. Suppose the adversary has picked some target ordering. We then presume that he will minimize detection probability up until some "transition point" at which time he will switch to time minimization. This transition point occurs when the adversary is close enough to all the remaining targets that, if he has been undetected so far, it is impossible for the guards to respond to an alarm and interrupt him at any of the remaining targets. The best path is different for each possible target ordering. So the real question becomes in what order should the targets be visited.

No polynomially-bounded-run-time algorithm is known for finding the shortest path in a network given k specified nodes to be visited. Dreyfus [1] discusses this problem and shows how it can be solved by solving an "open" traveling salesman problem on a related network. Taking advantage of the fact that in nuclear safeguards studies the number of specified targets, k, will always be small (say $k \leq 6$), we have avoided the traveling salesman (or Hamilton path) approach by simply minimizing the objective function over all k! possible target orderings along paths in the given facility digraph. Because our

branch-and-bound strategy can often avoid fully computing all k! possibilities, the O(k!) upper bound on run time is seldom achieved. Even if this upper bound is unavoidable, the fact that k is small makes the computation feasible. Our tests on realistic facility models have run in a matter of seconds (see Tables 1 and 2).

2. The Facility Model

The facility model used in POST is the same facility digraph used in ADPATH. This is a directed graph whose nodes represent locations on each side of each barrier penetration point and on each side of each target. The directed arcs represent physical paths from one location to another. Arcs crossing barriers represent penetration of the barriers. The arc from a target initiation node to a target completion node represents target sabotage. All the other arcs represent transit. Complete facility modeling details are given in [4, Sec. 2].

The weights on the digraph give important data concerning the barriers and alarm systems. Each arc has a given delay time and detection probability associated with the adversary's traversal of the arc. Each node has a given detection probability associated with the adversary's passing through the point, and the associated delay time is always zero.

A facility digraph typically has many possible targets, each one represented by two nodes and an arc. In each problem, the sabotage scenario usually involves only a proper subset of these, i.e., the "specified" sabotage targets. The user must give not only a list of the specified target completion nodes but also a list of respective guard-response times for each target and a list of possible adversary starting nodes.

3. The Path Problem

The problem is to find in what order and along what path the specified targets should be visited by the adversary so as to minimize his interruption probability. We assume that interruption will occur at a target if the adversary is detected while he is more than the guard-response time away from the target. That is, given such a detection, we assume that that target's guards are alerted and arrive at the target in time to prevent the completion of the sabotage of the target. If detection occurs while he is outside the response-time loci of

several targets, then of course he will be interrupted at the first one he visits. Therefore, the path we seek is a path in the weighted digraph which begins at any start node, passes through all the specified target completion nodes, minimizes detection probability from the start nodes up to a transition point, and minimizes time thereafter, the transition point being the earliest point lying not more than a guard-response time before each remaining target.

Any shortest-time path from the start nodes through the targets has a transition point found as follows. Back up along the path a guard-response time R_i before each target T_i and define that target's locus point P_i. The latest of these locus points is the path's transition point. For example, point P_3 is the transition point for the three-target path in Figure 1. P_3 is not more than a guard-response time from the remaining targets, and it is the earliest such point.

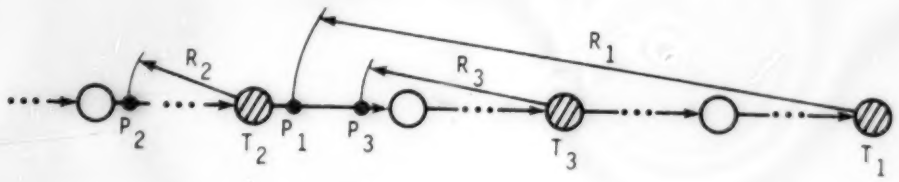

Figure 1. Transition Point P_3

4. The Solution

To solve this path problem we need to consider each possible target ordering. For each ordering, find a shortest-time path from the start nodes through the targets in order. Locate the transition point P for this path. There are three cases for where the transition point P will lie: (a) between two targets, T_m and T_ℓ, (b) between the start node and T_ℓ, or (c) before the start node.

In case (c) we are finished with this target ordering. The best path for this ordering is the shortest-time path we just calculated because there is no need to minimize detection probability. Suppose now that we have either case (a) or (b). Then that part of the best path (for this ordering) from T_ℓ onward has been determined. Let the transition point P lie Δ units of time before T_ℓ. What remains is to find a minimum-detection-probability path from either T_m or the start nodes (depending on which case we have) up to a time Δ away from T_ℓ together with a minimum-time path from the Δ-locus to T_ℓ. Let Q denote

the point where this path intersects the Δ-locus about T_ℓ, and notice that, although P and Q both lie Δ time units from T_ℓ, they may or may not be the same point. In case (b) we are finished with this target ordering. In case (a) we need to find a least-detection-probability path from the start nodes through each target in order up to T_m. For this ordering of the targets, the minimum-interruption-probability path is the path pieced together from: (i) the minimum-detection-probability segment from the start nodes through each target up to T_m; (ii) the transition segment, consisting of a minimum-detection-probability path from T_m to the transition point Q on the Δ-locus about T_ℓ and a minimum-time path from Q to T_ℓ; and (iii) the minimum-time segment from T_ℓ through the remaining targets. That path having the least detection probability value (up to its transition point) over all possible target orderings is the minimum-interruption-probability path we seek. Let Q* denote its transition point.

It should be clear that, if the transition point Q* for the result is in case (b), then segment (i) is empty and segment (ii) begins with a start node. Similarly, if Q* is in case (c), then both segments (i) and (ii) are empty and segment (iii) begins at a start node.

Should two target orderings yield paths with equal detection probability values, then the one with the smaller time value (from transition point to last target) is preferred. If they tie on this second value, the first one found is kept.

5. An Example

Let D be the digraph shown in Figure 2. Suppose we want to find a minimum-interruption-probability path in D through target nodes 1 and 2, which have respective guard-response times of $R_1 = 22$ and $R_2 = 23$. Let the adversary be an outsider so that the starting nodes are the off-site nodes 14 and 15.

There are only two target orderings. First, we consider a shortest-time path from 14 or 15 through 1 to 2. Such a path will have a node sequence of the form (...,3,1,4,2). The subpath (1,4,2) has a time value of 3 + 4 = 7 units. The locus point P_2, lying 23 time units before target node 2, must lie 23 - 7 = 16 units before target node 1, while the locus point P_1 lies 22 units before node 1. The later of these two, P_2, is the transition point for this path.

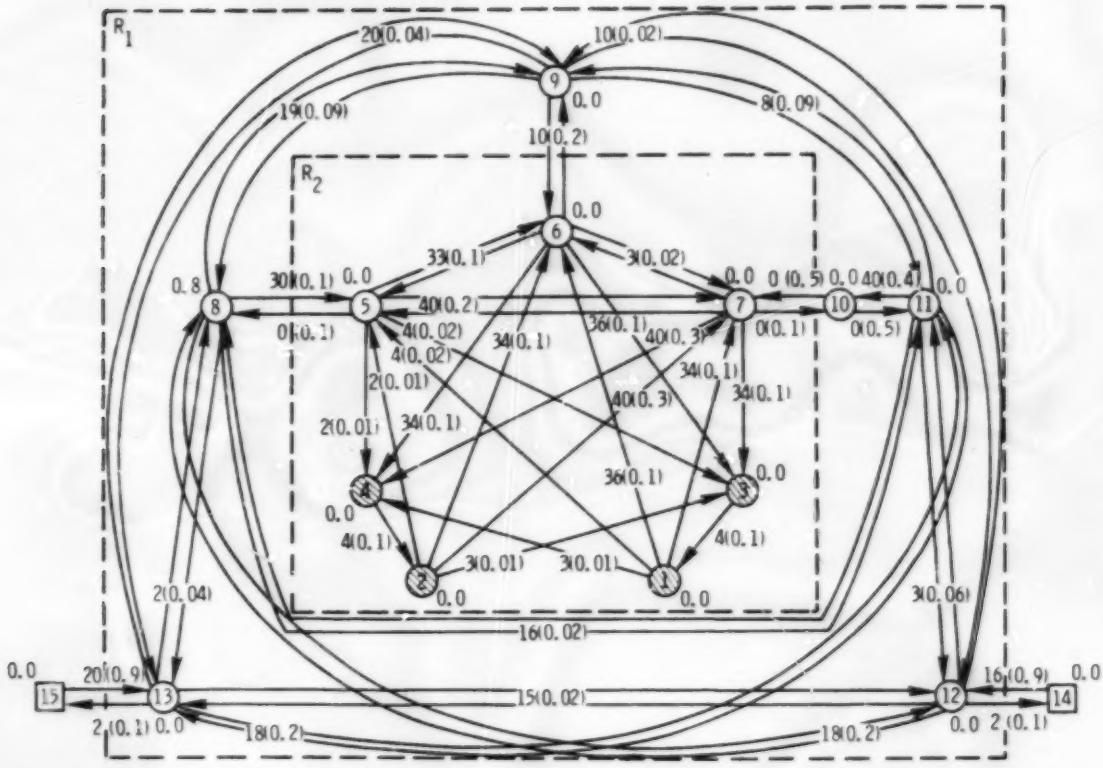

Figure 2. A Weighted Facility Digraph

The initial part of this shortest-time path up to P_2 does not necessarily minimize detection probability. Its only purpose is to help define $\Delta = 16$, the time from the transition point we seek to target node 1. The transition point Q, 16 units before node 1, will lie either 12 units back along one of the arcs (6,3) or (7,3) or else 8 units back along arc (8,5). It can be verified by inspection that the minimum-detection-probability path from either 14 or 15 up to any one of these three points is (14,12,9,6,Q), having a cumulative detection probability of

$$.9 \oplus .02 \oplus .2 \oplus \frac{24}{36}(.1) = .9 + .1(.02 + .98(.2 + .8(\frac{24}{36}(.1)))) \approx .9268266.$$

The fraction 24/36 results from the fact that Q lies 12 units back along the arc (6,3) whose time length is 36. Thus, the best path for

this ordering is (14,12,9,6,3,1,4,2) having a detection probability value (from 14 to Q) of .9268266 and a time value (from Q to 2) of 23.

Now consider a shortest-time path from the start nodes through 2 to 1. The only important part of this path is (...,4,2,3,1). The subpath (2,3,1) has a time value of 3 + 4 = 7 units. The locus point P_2 lies 23 units before node 2, while the locus point P_1 lies 22 - 7 = 15 units before node 2. Therefore, the transition point Q for this ordering lies Δ = 15 units before node 2 and must be 15 - 4 = 11 units before node 4. Again, there are three possibilities for Q: 11 units back along (6,4) or (7,4) or 9 units back along (8,5). The best path for this ordering is (14,12,9,6,4,2,3,1) having a detection probability value of

$$.9 \oplus .02 \oplus .2 \oplus \frac{23}{34}(.1) = .9 + .1(.02 + .98(.2 + .8(\frac{23}{34}(.1)))) \approx .9269035$$

and a time value of 22.

Consequently, the minimum-interruption-probability path is (14,12,9,6,3,1,4,2) having a value of .9268266.

6. Calling POST

Listings of POST may be found on a microfiche card inside the back cover of this report. The first listing is in FLECS (FORTRAN Language with Extended Control Structures), the language in which POST was developed and which is easiest to read. The FORTRAN code produced by the FLECS preprocessor is also given.

The call list for POST is

```
POST(N,N1,N2,N3,NO,NA,NSTRT,NTSET,DIMIN,
     START,DPN,T,DPA,JNDX,IPTR,TSET,RT,
     DIMOUT,IERR,PTHLNG,SQSPTH,SQSDP,SQST,
     IWORK,WORK).
```

The arguments have the following meanings.

Input:

N - total number nodes (N = N1 + N2 + N3);

N1 - number of target nodes (target completion nodes are numbered first and all other target nodes are numbered second);

N2 - number of barrier nodes (barrier nodes are numbered third);

N3 - number of boundary nodes (internal boundary nodes are numbered fourth and off-site nodes are numbered last);

NO - number of off-site nodes;

NA - number of arcs;

NSTRT - number of starting nodes;

NTSET - number of targets in the specified subset, NTSET $\geq$ 2;

DIMIN - amount of space allowed for SQSPTH (a guess which may be computed by (the average number of nodes in a start-to-target or target-to-target path)*(2*NTSET + 2) + N);

START - an integer array of starting nodes, dimensioned NSTRT;

DPN - a real array of detection probabilities for the nodes, dimensioned N;

T - a real array of arc times in sparse matrix form, dimensioned NA;

DPA - a real array of arc detection probabilities in sparse matrix form, dimensioned NA;

JNDX - an integer array of column numbers for the sparse matrix form of T and DPA, dimensioned NA;

IPTR - an integer array of locations of the last entry for each row in the sparse matrix form of T and DPA, dimensioned N;

TSET - an integer array of target completion nodes for those targets to be included in the path, dimensioned NTSET;

RT - a real array of target response times corresponding to the targets in TSET, dimensioned NTSET.

Notice that T, DPA, JNDX, and IPTR provide an economical storage scheme for the two arc-weight matrices. T and DPA are the finite, off-diagonal, entries from these matrices listed row by row. JNDX is a list of the column numbers for each of these entries, and IPTR points to the last entry from each row of the matrices. The storage required is 3NA + N instead of $2N^2$.

Output:

The solution consists of one path, stored in SQSPTH, and its two numerical values SQSDP (detection probability) and SQST (time). Other numbers computed for output indicate how much space is actually needed

in the array SQSPTH, whether or not any errors occurred, and how many nodes are in the output path.

DIMOUT - the actual space needed for all of the uses of SQSPTH;
IERR - an error status indicator,
= 0 means there were no errors,
= 1 means that DIMIN was not big enough and the problem must be rerun with DIMIN set equal to DIMOUT,
= 2,...,12 means that certain input variables had inadmissable values (see listing) which must be corrected before rerunning the problem.

The above variables are always meaningful. The following output is meaningful only when IERR = 0. Therefore, the user must always test for IERR = 0 before accepting any output path data.

PTHLNG - actual number of nodes in the output path including the repetition of the node at or immediately preceding the transition point;
SQSPTH - an integer array of nodes in the output sequential sabotage path including the repetition of the node at or just before the transition point, dimensioned DIMIN;
SQSDP - sequential sabotage path detection probability accumulated from start node to transition point;
SQST - sequential sabotage path time accumulated from transition point to last target.

SQSPTH not only contains the output path but also all other arrays of unpredictable length. This accounts for the uncertainty in its dimension, DIMIN, and the difference between its required length, DIMOUT, and the length, PTHLNG, of the output path. When IERR = 0, the output path is listed as a node sequence in SQSPTH(1),SQSPTH(2),..., SQSPTH(PTHLNG).

Mention has already been made of the repetition of the node at or immediately preceding the transition point. This helps the user roughly locate the transition point. Whenever the transition point lies before the start node, however, we simply repeat the start node. In this case, SQSDP = 0. and SQST is the actual time from the start node to the end of the path.

Work Arrays:

IWORK - an integer array of working storage, dimensioned NTSET(3N + 6) + 2N;

WORK - a real array of working storage, dimensioned NTSET(2N + NTSET + 4).

As indicated above, all work arrays of unpredictable length have been stored in SQSPTH so that only two arguments, DIMIN and DIMOUT, are needed to check for adequate storage allocation.

7. Solving the Example with POST

The weighted digraph in Figure 2 has 4 target nodes, 7 barrier nodes and 4 boundary nodes (2 of which are off-site nodes) for a total of 15 nodes and 54 arcs. Consequently, we set

N = 15 N1 = 4 N2 = 7 N3 = 4 NO = 2 NA = 54.

The start nodes are the off-site nodes, so

NSTRT = 2 and START = (14,15).

The specified target completion nodes are 1 and 2, having respective response times of 22 and 23, so that

NTSET = 2 TSET = (1,2) RT = (22,23).

If we guess that there will be about five nodes in an average target-to-target or start-to-target segment, then we should set

DIMIN = 5(2NTSET + 2) + N = 45.

The only node having a positive jump in detection probability is node 8, and so

DPN = (.0,.0,.0,.0,.0,.0,.0,.8,.0,.0,.0,.0,.0,.0,.0).

The sparse matrix form for the arc times, T, and the arc detection probabilities, DPA, is

```
T = ( 3.,4.,36.,34.|3.,2.,34.,40.|4.|4.|
      4.,2.,33.,40.,0.|36.,34.,33.,3.,10.|
      34.,40.,40.,3.,0.|30.,19.,16.,18.,2.|
      10.,19.,8.,10.,20.|0.,0.|16.,8.,40.,
      3.,18.|18.,10.,3.,15.,2.|2.,20.,18.,
      15.,2.|16.|20.),

DPA = (.01,.02,.1,.1|.01,.01,.1,.3|.1|.1|
       .02,.01,.1,.2,.1|.1,.1,.1,.02,.2|
       .1,.3,.2,.02,.1|.1,.09,.02,.2,.04|
       .2,.09,.09,.02,.04|.5,.5|.02,.09,.4,
       .06,.2|.2,.02,.06,.02,.1|.04,.04,.2,
       .02,.1|.9|.9),

JNDX = (4,5,6,7|3,5,6,7|1|2|
        3,4,6,7,8|3,4,5,7,9|
        3,4,5,6,10|5,9,11,12,13|
        6,8,11,12,13|7,11|8,9,10,
        12,13|8,9,11,13,14|8,9,11,
        12,15|12|13),

IPTR = ( 4,8,9,10,15,20,25,30,35,37,42,47,52,53,54).
```

Calling POST with this input results in the output

```
DIMOUT = 44   IERR = 0   PTHLNG = 9
SQSPTH = (14,12,9,6,6,3,1,4,2)
 SQSDP = .9268266   SQST = 23.
```

This means that no errors occurred and that the minimum-interruption-probability path for sequential sabotage is (14,12,9,6,3,1,4,2) having the transition point either at node 6 or else along arc (6,3). Backing up 23 time units from the end of the path shows that it is the latter case, the transition point lies two-thirds of the way from 6 to 3. The probability of the adversary's being interrupted along this path is .9268266.

8. Description of POST

POST uses ADPATH to do all of its pathfinding. Recall that ADPATH accepts the same facility digraph and starting nodes as does POST. However, for sabotage problems, ADPATH requires a list of sabotage

targets and corresponding response times from which it produces a single-target minimum-interruption-probability path for each target in the list. Each of these paths minimizes detection probability from the start nodes up to the response-time locus about each target and minimizes time from the locus (transition point) to the target.

POST may be divided into three sections: finding various path segments; computing numerical values for best paths associated with each possible target ordering, while saving the best one found so far; and piecing together the node sequence for the optimum path.

First, shortest-time paths to each target from all other nodes are found. This is done by calling ADPATH with START for the start nodes, TSET for the target nodes, and infinite (10^{38}) response times. The normal output paths from ADPATH are least-time paths from START to each target. However, in one of its work arrays, ADPATH has also stored least-time paths from all nodes to each target. This information is also extracted from ADPATH. (Since ADPATH's theft problem capabilities are not used in this context, it is storage efficient to tell ADPATH that there are no off-site nodes so that no space is devoted to certain theft-path arrays associated with paths to off-site.)

Next, POST finds minimum-detection-probability paths: (a) from START to each target and (b) from each target to every other target. In (a), ADPATH is given START for the start nodes, TSET for the targets, and zero response times so that time-minimization portions of the paths are empty. In (b), ADPATH is called once for each I = 1,2,..., NTSET, with TSET(I) as the single start node, TSET as the targets, and zero response times.

In POST's second section, each possible ordering of the targets is considered. The given target set, TSET, is permuted in all NTSET! ways by Nijenhuis' and Wilf's subroutine NEXPER [5]. For each permutation, POST steps backwards through the targets along a least-time path accumulating the target-to-target times until the segment containing the transition point is found. If the transition point lies on or before a start node, then the best path for this permutation is this shortest-time path, having a detection probability value of zero. Otherwise, POST must find a least-detection-probability path from START through the targets in order up to a Δ-locus about T_ℓ, where T_ℓ is the target at the end of the transition segment and Δ is the time from the transition point to T_ℓ. The transition segment may begin either at

START (if T_ℓ is the first target in this permutation) or else at the target T_m preceding T_ℓ in this permutation. The transition segment is simply the normal output path from ADPATH given either START or T_m as the start node(s), T_ℓ as the target, and Δ as the response time. The transition segment's detection probability is then added to the accumulated value of all previous minimum-detection-probability segments, if any, from START through each target in the permutation up to T_m. This value is compared with the least value found so far, and the path having the smaller detection probability value is saved along with its two numerical values. When the paths have identical detection probability values, the one with the smaller time is saved. When both values are identical, the first path found is saved. The path saved after all target permutations have been considered is the path we seek.

The reader should be aware of a branch-and-bound strategy that is actually used in the above minimization process. Before ADPATH is called to find a transition segment, the accumulated detection probability for the preceding segments is computed and compared with the best path detection probability found so far. If this target permutation's partial detection probability is greater than the best path probability found so far, then POST avoids calling ADPATH and proceeds directly to the next target permutation. So ADPATH is not necessarily called NTSET! times.

The third section of POST uses predecessor and successor arrays saved from the previous calls of ADPATH to form the output path's node sequence, repeating either the start node or else the [redacted] node at or immediately preceding the transition point. Where we mention "saving a path" earlier, we actually save the corresponding target permutation as well as indices indicating the transition segment. From this and the predecessor and successor arrays for the various path segments, the output path is constructed.

This code was developed for the CDC 7600, but it should be very easy to move to other machines. The FORTRAN version of POST meets ANSI standards. Only two constants are related to machine word length, and they occur only in easily found DATA statements at the beginning of certain subroutines. If 10^{38} (infinity in the code) is too large for the new machine, then it should be replaced in the DATA statements of subroutines POSTAD, FIND, and SPATHF by the largest representable power of 10. If the machine's real number has about d significant digits and

d>15, then the real constant 34 in a DATA statement of subroutine FIND should be replaced by the integer part of 2.3*d.

9. Subroutine Performance

The storage required by POST is dominated by its arrays,

$$\text{array storage} = 3NA + 5N(NTSET + 1) + NTSET(NTSET + 12) + NSTRT + DIMOUT.$$

The dominant term here tends to be 3NA when NTSET = 2 or 3 and 5N·NTSET when NTSET $\geq$ 4.

Table 1 summarizes POST's performance on problems of various sizes. Although the example problem of Sections 5 and 7 is not in this table, it is most like Problem 3.

Table 1

POST Performance

Problem	Nodes N	Arcs NA	Targets NTSET	Start Nodes NSTRT	SQSPTH Dimens. DIMOUT	Array Storage (decimal)	Run Time CDC 7600 (seconds)
1	16	43	3	2	51	547	0.022
2	16	43	3	2	51	547	0.022
3	20	64	2	2	73	595	0.025
4	16	43	3	2	51	547	0.022
5	14	39	2	1	48	404	0.017
6	28	92	4	2	113	1155	0.067
7	34	97	2	8	53	890	0.038
8	38	117	2	8	57	1014	0.037
9	92	315	4	14	133	3456	0.196
10	164	1172	3	4	225	7070	0.652
11a	310	1592	2	5	351	9810	0.764
11b	310	1592	3	5	393	11,419	1.174
11c	310	1592	4	5	439	13,034	3.421
11d	310	1592	5	5	453	14,619	2.840
11e	310	1592	6	5	489	16,228	10.720

Problem 11 concerns a realistic reactor site model. Target sets of cardinality 2 through 6 were specified for this model to give some

feeling for the variation of storage and run time with various values of NTSET. Notice that the array storage increases monotonely as NTSET increases but the run time does not. The five-target problem ran faster than the four-target problem. These target sets are not nested, i.e., the five-target set does not include the four-target set. A little reflection will convince the reader that it is quite possible for a best five-target ordering to be found early, in which case many calls to ADPATH are avoided by POST's branch-and-bound strategy, while the opposite can happen for a four-target problem.

This observation raises the question of how run time depends on the order in which the various target permutations are considered. Without thinking, we obtained the results of Table 1 by simply listing the nodes in TSET in order of increasing node number. This given order in TSET is the first target ordering considered, and all others are derived from it by NEXPER [5]. Obviously, we will obtain different run times if we give TSET in a different order because the entire sequence of target permutations will be different and so will be the number of calls to ADPATH.

Table 2 shows the possible range of variation in run time with different given target orderings for the realistic model in Problem 11 when $3 \leq$ NTSET ≤ 6. For $3 \leq$ NTSET ≤ 5, all NTSET! possible orderings were used as given orderings. For NTSET = 6, only 36 samples were run, including the ordering known to be best and many expected to be very poor in run time.

Table 2

Range of POST Run Times with Different Given Target Orderings

Problem	No. of Targets NTSET	CDC 7600 Run Time (seconds) Minimum	Maximum
11b	3	1.2	1.4
11c	4	3.2	3.7
11d	5	2.8	5.2
11e	6	10.1	17.1

Clearly, it would be desirable to give POST the best target ordering since this would minimize its run time. However, we cannot know what this is before running POST. The next best thing to do is to look

at the graph a moment and guess at a reasonable target ordering. At least if one avoids giving POST an obviously bad ordering, such as one which oscillates between target clusters, there will be a tendency for the run time to fall in the lower half of the range.

References

[1] S. E. Dreyfus, "An Appraisal of Some Shortest-Path Algorithms," Operations Res., 17:395-412, 1969.

[2] B. L. Hulme and D. B. Holdridge, SPTH3: A Subroutine for Finding Shortest Sabotage Paths, SAND77-1060 (Albuquerque: Sandia Laboratories, July 1977).

[3] B. L. Hulme, MINDPT: A Code for Minimizing Detection Probability up to a Given Time Away from a Sabotage Target, SAND77-2039 (Albuquerque: Sandia Laboratories, December 1977).

[4] B. L. Hulme and C. A. Morgan, ADPATH: An Adversary Path Subroutine, SAND79-1205 (Albuquerque: Sandia Laboratories, July 1979).

[5] A. Nijenhuis and H. S. Wilf, Combinatorial Algorithms (New York: Academic Press, 1975).

DISTRIBUTION:

U.S. Nuclear Regulatory Commission (290 copies for RS)
Division of Document Control
Distribution Services Branch
7920 Norfolk Branch
Bethesda, MD 20014

U.S. Nuclear Regulatory Commission
MS 88155
Washington, DC 20555
Attn: M. Fadden

U.S. Nuclear Regulatory Commission (2)
MS 1130SS
Washington, DC 20555
Attn: R. Robinson

Los Alamos Scientific Laboratory
Attn: G. R. Keepin, R. A. Gore, E. P. Schlonka, D. G. Rose
Los Alamos, NM 87544

Allied-General Nuclear Services
Attn: G. Molen
P.O. Box 847
Barnwell, SC 29812

Lawrence Livermore Laboratory
University of California
P.O. Box 808
Attn: A. J. Poggio
Livermore, CA 94550

Pritsker and Associates, Inc.
P.O. Box 2413
Attn: F. H. Grant
West Lafayette, IN 47906

Union Carbide Corporation
Nuclear Division
Bldg. 7601
Attn: D. Swindle
Oak Ridge, TN 37830

1230 W. L. Stevens, Attn: R. E. Smith, 1233
1700 W. C. Myre
1710 V. E. Blake, Attn: M. R. Madsen, J. W. Kane
1716 R. L. Wilde
1720 C. H. Mauney, Attn: J. D. Williams, 1729
1750 J. E. Stiegler, Attn: M. J. Eaton
1754 I. G. Waddoups, Attn: J. L. Todd, 1754
1758 C. E. Olson, Attn: G. A. Kinemond, 1758
1758 D. D. Boozer
1760 J. Jacobs, Attn: M. N. Cravens, J. M. deMontmollin, 1760A
1761 T. A. Sellers, Attn: A. E. Winblad, J. L. Darby, 1761
1762 H. E. Hansen
1765 D. S. Miyoshi
2614 C. A. Morgan
4400 A. W. Snyder
4410 D. J. McCloskey
4414 G. B. Varnado

DISTRIBUTION (Continued)

4416 L. D. Chapman
4416 K. G. Adams
4416 J. A. Allensworth
4416 H. A. Bennett
4416 D. Engi
4416 L. M. Grady
4416 C. P. Harlan
4416 R. D. Jones
4416 M. T. Olascoaga
4416 C. J. Pavlakos
4416 D. W. Sasser
4416 D. R. Strip
5000 J. K. Galt
5600 D. B. Shuster, Attn: A. A. Lieber, 5610, M. M. Newsom, 5620
R. C. Maydew, 5630
5640 G. J. Simmons, Attn: R. J. Thompson, 5641
L. F. Shampine, 5642
5642 B. L. Hulme (4)
8266 E. A. Aas
3141 T. L. Werner (5)
3151 W. L. Garner (3)
For: DOE/TIC (Unlimited Release)
3172-3 R. P. Campbell (25)
For NRC for NTIS

THIS PUBLICATION CONTAINED
MATERIAL THAT DID NOT LEND
ITSELF TO MICROFILMING
CRITERIA.